APPLICATION

FOR

UNITED STATES LETTERS PATENT

APPLICANT NAME: C. N. Kline

TITLE: METHOD AND APPARATUS FOR SCHEDULING THE PERFORMANCE OF MAINTENANCE TASKS TO MAINTAIN A SYSTEM ENVIRONMENT

DOCKET NO.: END920030058US1

INTERNATIONAL BUSINESS MACHINES CORPORATION

Certificate of Mailing Under 37 CFR 1.10

I hereby certify that, on the date shown below, this correspondence is being deposited with the United States Postal Service in an envelope addressed to the Commissioner for Patents, P.O. Box 1450, Alexandria, VA 22313-1450 as "Express Mail Post Office to Addressee"

"Express Mail" Label No.: EV 342658825 US

On: 8/6/2003

BETHANY J. FITZPATRICK

Typed or Printed Name of Person Mailing Correspondence

Bethany J. Fitzpatrick 8/6/03

Signature **Date**

METHOD, APPARATUS AND PROGRAM STORAGE DEVICE FOR SCHEDULING THE PERFORMANCE OF MAINTENANCE TASKS TO MAINTAIN A SYSTEM ENVIRONMENT

BACKGROUND OF THE INVENTION

1. Field of the Invention.

This invention relates in general to maintaining a computer system, and more particularly to a method, apparatus and program storage device for scheduling the performance of maintenance tasks to maintain a system environment.

2. Description of Related Art.

The defining environment for application development is changing. E-business applications are being used to leverage the Internet as a platform for building and integrating applications over a network environment. Further, application servers are moving from a processor-based operating system to an Internet-based operating system or network computing. However, this term may be somewhat ambiguous. Nevertheless, this term is increasingly being used to refer to virtual applications that are assembled from many different components that run on many machines across a network as if they were a single system. Components can range from entire centralized applications to single modules of a larger distributed application. Most important, it is becoming clear that only virtual applications can deliver the flexibility to meet many of the lead-edge needs of today's corporate information systems. Accordingly, these application servers are being used to integrate applications, business processes, and data. To leverage

investments, businesses are turning to open standard based infrastructures to allow solutions to be crafted based on the best products for an application, process, etc. rather being locked into a single-vendor solution.

However, such an open standard integrated solution throughout the enterprise requires tools and systems to accomplish the integration, connectivity, modeling, monitoring and management functions. People, processes, information, and systems must be integrated throughout the enterprise. Connectivity refers to connecting applications and systems across a company and to partners and customers. Modeling includes the ability to model and simulate business processes to graphically represent the flow of work across people and application systems. Monitoring is provided by tracking business processes as they execute in applications and systems across the enterprise. Management of the enterprise demands visualization of immediate operational results of business processes. This critical knowledge enables review of business and system processes over a period of time so that bottlenecks and problem areas can be identified and corrected.

In order to connect processes and to transform data, a diverse collection of functions must be provided. This diverse collection of functions is referred to as middleware. The term middleware is an inclusive term that encompasses many disparate functions that do not easily fit within other architectural components. Thus, middleware may be considered an aggregation of distinct subcomponents. Middleware provides application services that were once written into applications. However, these services today are provided in an independent infrastructure layer. Middleware enhances

application integration by providing uniform mechanisms to bridge ole and new technologies, or by enabling dissimilar elements to work together.

One type of middleware is message-oriented middleware. Message-oriented middleware allows application programs that may be distributed across similar or dissimilar platforms and/or network protocols to exchange data with each other using messages and queues. In order to maintain a clean and efficient environment, certain maintenance procedures must be run regularly. Such maintenance tasks, however, are not easily performed by built-in utilities. In fact some important tasks cannot be performed at all by built-in utilities. Thus, the system administrator is left with the task of determining how to best carry out these tasks. Still, not all administrators are skilled in programming such tasks. Moreover, administrators may not know the optimum time to run maintenance tasks. For example, maintenance tasks may be scheduled to run too often. Because such maintenance tasks may be resource intensive, the performance of the server may suffer. Alternatively, maintenance tasks may not be scheduled often enough and therefore the system may not be operating efficiently.

It can be seen then that there is a need for a method, apparatus and program storage device for scheduling the performance of maintenance tasks to maintain a system environment.

SUMMARY OF THE INVENTION

To overcome the limitations in the prior art described above, and to overcome other limitations that will become apparent upon reading and understanding the present specification, the present invention discloses a method, apparatus and program storage device for scheduling the performance of maintenance tasks to maintain a system environment.

The present invention solves the above-described problems by providing a way for administrators can skillfully maintain servers. The present invention determines when to run maintenance operations based on predetermined system criteria.

A method in accordance with an embodiment of the present invention includes monitoring a parameter for a computer system to detect a need to perform at least one maintenance task and performing at least one maintenance task when the monitoring detects the need to perform at least one maintenance task or at least once within a predetermined period.

In another embodiment of the present invention, a system for scheduling the performance of maintenance tasks to maintain a system environment is provided. The system includes a maintenance tool for providing resources for performing at least one maintenance task and a maintenance scheduling device for monitoring a parameter for a computer system to detect a need to perform at least one maintenance task and causing the maintenance tool to perform at least one maintenance task when the maintenance scheduling device detects the need to perform at least one maintenance task or at least once within a predetermined period.

In another embodiment of the present invention, another system for scheduling the performance of maintenance tasks to maintain a system environment is provided. This system includes means for providing resources for performing at least one maintenance task and means for monitoring a parameter for a computer system to detect a need to perform at least one maintenance task and causing the means for providing resources to perform at least one maintenance task when the means for monitoring detects the need to perform at least one maintenance task or at least once within a predetermined period.

In another embodiment of the present invention, a program storage medium tangibly embodying one or more programs of instructions executable by the computer to perform a method for scheduling the performance of maintenance tasks to maintain a system environment is provided. The method includes monitoring a parameter for a computer system to detect a need to perform at least one maintenance task and performing at least one maintenance task when the monitoring detects the need to perform at least one maintenance task or at least once within a predetermined period.

These and various other advantages and features of novelty which characterize the invention are pointed out with particularity in the claims annexed hereto and form a part hereof. However, for a better understanding of the invention, its advantages, and the objects obtained by its use, reference should be made to the drawings which form a further part hereof, and to accompanying descriptive matter, in which there are illustrated and described specific examples of an apparatus in accordance with the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

Referring now to the drawings in which like reference numbers represent corresponding parts throughout:

Fig. 1 illustrates a distributed computing environment according to an embodiment of the present invention;

Fig. 2 illustrates a middleware layer infrastructure according to an embodiment of the present invention;

Fig. 3 illustrates a system for scheduling the performance of maintenance tasks to maintain a system environment according to an embodiment of the present invention;

Fig. 4 illustrates examples of management tools according to an embodiment of the present invention;

Fig. 5 illustrates a flow chart of the maintenance program execution according to an embodiment of the present invention;

Fig. 6 illustrates a detailed flow chart for the maintenance and recovery task operations according to an embodiment of the present invention; and

Fig. 7 illustrates the dynamic execution of the maintenance utility according to an embodiment of the present invention.

DETAILED DESCRIPTION OF THE INVENTION

In the following description of the embodiments, reference is made to the accompanying drawings that form a part hereof, and in which is shown by way of illustration the specific embodiments in which the invention may be practiced. It is to be understood that other embodiments may be utilized because structural changes may be made without departing from the scope of the present invention.

The present invention provides a method, apparatus and program storage device for scheduling the performance of maintenance tasks to maintain a system environment. Administrators use the present invention to skillfully maintain servers. When to run maintenance operations is determined based on predetermined system criteria.

Fig. 1 illustrates a distributed computing environment 100 according to an embodiment of the present invention. Fig. 1 illustrates three computing domains 110-114. The three computing domains 110-114 include at least one server 120 and at least one client 122 networked to the server 120. The domains 110-114 are coupled through a network 130. Applications running in each of the domains 110-114 are usually built using centralized approaches, wherein the business rules and data that comprise the application reside on a single mainframe or network server, and lack ways to participate as components on the network. To prevent isolation between the domains 110-114, middleware services 140 are used to provide uniform mechanisms to bridge the different technologies and to enable dissimilar elements to work together. Middleware 140 comprises computer software that runs on a computer in at least one of the clusters 110-114.

Fig. 2 illustrates a middleware layer infrastructure 200 according to an embodiment of the present invention. In Fig. 2 the middleware layer 210 provides an independent infrastructure layer that sits between applications 212, networks 214, databases 216, and distributed services communications mechanisms 218. The middleware layer 210 enhances application integration by providing uniform mechanisms to bridge between the applications 212, networks 214, databases 216, and distributed services communications mechanisms 218. Management of the infrastructure is facilitated by the middleware layer 210 by allowing for the centralized management of application services that are shared by many applications. The middleware layer 210 also reduces the costs and complexity of implementing change in infrastructure services by enabling changes to be made across many applications, rather than to each application that uses a service.

One type of middleware services that is used to ensure that the applications and other functions are running smoothly is maintenance utilities. For example, in a distributed computing environment, certain maintenance procedures must be run regularly to ensure the system is operating properly. For example, servers in a computing environment must be monitored to prevent crashes and/or to facilitate peak performance. Such maintenance tasks, however, are not easily performed by built-in utilities. In fact some important tasks cannot be performed at all by built-in utilities. Thus, the system administrator is left with the task of determining how to best carry out these tasks. Still, not all administrators are skilled in programming such tasks. Moreover, administrators may not know the optimum time to run maintenance tasks. Thus, the present invention

provides a maintenance tool to provide an administrator the ability to determine when settings changed as well as the option of restoring old settings through saved files.

Fig. 3 illustrates a system 300 for scheduling the performance of maintenance tasks to maintain a system environment according to an embodiment of the present invention. In Fig. 3, a central processing unit 310 may access memory 312 and runs an operating system 314. The operating system 314 may access application 320. The application 320 includes instances 330-334. Master configuration file 344 identifies each of the application instances configured on the system, as well as default settings for each instance. Each instance 330-334 may include an application instance-specific configuration file 340 and application instance transaction logs 342. .

The present invention includes a maintenance scheduling tool 350 to schedule maintenance tasks. Maintenance tasks are providing via the maintenance tools 352. The maintenance tools 352 maintain a maintenance log file 354. The maintenance tools 352 are part of a larger suite of tools created to provide daily maintenance routines, as well as routines that copy settings to an archive location for later point-in-time recovery, if necessary. A script is provided via the maintenance scheduling tool 350 to determine how often the maintenance tools 352 are to be performed. For example, administration setup maintenance may need to be performed nightly on all servers. However, some servers that are heavily used may need maintenance run more often than just once a day. To statically schedule run times, however, may be too little or too much.

In one embodiment of the present invention, the script provided in the maintenance scheduling tool 350 to control the maintenance tools 352 is a korn-shell

script. A korn-shell script is a program that reads textual commands from the user or from a file, converts them into operating system commands, and executes them. A wrapper 356 sits on top of the script in the maintenance scheduling tool 350 to dynamically determine whether the maintenance tools 352 should be run. For example, the wrapper 356 may look at regular intervals such as every 30 minutes. The wrapper 356 will cause the maintenance tools to be run at least every 24 hours or other preset period, but also self-regulates in case log filesystems are in danger of filling up. Thus, the wrapper 356 in the maintenance scheduling tool 350 monitors server conditions and runs the maintenance tools 352 whenever prudent. However, the present invention is not meant to be limited to a particular period or pattern. In this manner, the utilities 352 function as a "maintenance on demand" package of tools. If the maintenance tools 352 are run too often thereby indicating some sort of problem on the server, an alert is generated and the administrator is paged, for example, via email 360.

Some commands may be executed directly within the interpreter itself, e.g., setting variables or control constructs, while others may cause it to load and execute other files. Unix's command interpreters are known as shells. The maintenance tools 352 employ several other external programs to complete its operation.

The cron scheduling utility 370 is an operating system scheduler that monitors for a set time and then executes a predetermined command. For example, the maintenance tools 352 may be scheduled to complete once each night and then log all operations to an output log. The maintenance tools 352 then may email 360 the administrator whenever it doesn't complete successfully. This provides administrators with the ability to manage

problems before they grow and take down the server. For example, the maintenance tools may be used to clear unneeded log space.

Log files 358, such as the channel exit log file, are pruned to keep from growing too large. Log files 358 may have a preset size, and once full, existing files are copied to a backup file. Any "error" output may be emailed to the administrator for review. Also, all program activity may be logged to a file 358 for review; and to track whether maintenance was started manually or via a cron scheduling utility.

A management console 380 provides an administrator access to and control of the maintenance tools 352 and scheduler 350. The management console 380 may include a display 382 for displaying status or other data provided by the maintenance tools 352.

Fig. 4 illustrates examples of management tools 400 according to an embodiment of the present invention. Fig. 4 illustrates the save command 410, which is used to dump all configuration information to a text file for reuse 412. The last two iterations may be kept actively, with older versions retained as archives. Another tool is the built-in checkpoint command 420 for all running queue managers that use linear logging. (A queue manager is an application instance for asynchronous messaging applications such as IBM WebSphere MQ). The built-in checkpoint command 420 causes all objects to be written to log for immediate recovery if corrupted 422.

A clean log command 430 may be provided for all running queue managers using linear logging. The clean log script 430 looks in the error logs for the latest transaction logs that are needed for full recovery 432. The clean log script 430 will then take older logs and zip them up to conserve space. A prune command 440 may be provided to

prune old compressed logs from clean logs that are no longer needed. The prune command 440 keeps linear logging queue managers from taking over the filesystem and filling it up 442. If linear logging queue managers are provided and do not run this, the filesystem will eventually fill up and take the queue manager down, causing an outage. The prune utility 440 provides administrators with the ability to roll back to an earlier point in time, but also keeps file systems free of obsolete files.

The save authorizations script 450 is a command that should be run periodically for all running queue managers. The save authorization script 450 provides recovery of application-based authorizations in case a queue manager needs to be rebuilt, or if moved to another box 452. Such scripts may be kept in a specified log file. For example, ten of the authorization scripts may be kept to provide 2.5 months' ability to recover should problems be encountered. An archive-old-configuration-files script 460 provides logs beyond the standard two configuration files for each queue manager. The archive-old-configuration-files command 460 takes old configuration files periodically, and copies them to an archive directory 462. For example, if taken twice weekly, 50 of these archive files may be kept to provide six full months recovery ability. The archive-old–configuration-files feature is especially helpful for queue managers whose definitions (configuration) change regularly, or where non-standard administrators may have access to alter objects, and a record of it needs to be kept.

Those skilled in the art will recognize that the present invention is not meant to be limited to the maintenance tasks illustrated above with reference to Fig. 4. Rather, embodiments in accordance with the present invention may run any type of maintenance

tools based on system criteria. Further, by combining maintenance tasks, a maintenance scheduling tool according to embodiments of the present invention may provide a comprehensive maintenance routine. For example, the combination of the save file command 410 and the save authorization script 450 allows a complete snapshot of activity that can later be used to fix problems or restore configurations.

Fig. 5 illustrates a flow chart 500 of the maintenance program execution according to an embodiment of the present invention. In Fig. 5, the tool determines the operating-system type and tailors execution to the specified environment 510. Application instances are then retrieved 512. For each instance, maintenance and recovery-preparation tasks are performed 514. Errors during execution of the maintenance and recovery-preparation tasks are monitored 516. A determination is made whether errors occurred 520. If errors occurred 532, the administrator is alerted of the error condition 540. If not 534, the maintenance program terminates 550. All execution activities are logged for later review 542.

Fig. 6 illustrates a detailed flow chart 600 for the maintenance and recovery task operations (e.g., block 514 of Fig. 5) according to an embodiment of the present invention. In Fig. 6, the maintenance and recovery task is initiated 610. A transaction log type is obtained 612. The application instance version and status is also obtained 614. A determination is made whether the instance is running 620 and whether the log type is linear or circular 630, 640. If the instance is not running 622 and the system is circular 632, the archived configuration and authorization files are cleaned-up 680 and the system returns to check for execution errors 690.

If the instance is running 624 and the system is circular 642, the current configuration is exported to an archive 650. The current authorizations are also exported to an archive 660. Then, the archived configuration and authorization files are cleaned-up 680 and the system returns to check for execution errors 690.

If the instance is running 624 and the system is linear 644, the transaction checkpoint is recorded 670, e.g., the transaction log files are updated for pruning. The old transaction logs are zipped and removed 672. Because the instance is running, the current configuration is exported to an archive 650. The current authorizations are also exported to an archive 660. Then, the archived configuration and authorization files are cleaned-up 680 and the system returns to check for execution errors 690.

If the instance is not running 622 and the system is linear 634, the old transaction logs are zipped and removed 674. Because the instance is not running 622, the archived configuration and authorization files are merely cleaned-up 680 and the system returns to check for execution errors 690. Those skilled in the art will recognize that the blocks deciding whether the log type is linear or circular 630, 640 may be the same process, and in a like manner so may the blocks where old transaction logs are zipped and removed 672, 674.

Furthermore, those skilled in the art will recognize that the present invention is not meant to be limited to the maintenance tasks illustrated above with reference to Fig. 6. Rather, embodiments in accordance with the present invention may run any type of maintenance tools based on system criteria. Examples of additional maintenance tasks that may be performed according to the maintenance scheduling tool of the present

invention include, but are not limited to, providing database consistency checks, ensuring database compaction, performing full-text index generation, performing view recalculations, etc.

Fig. 7 illustrates the dynamic execution of the maintenance utility 700 according to an embodiment of the present invention. In Fig. 7, the cron scheduling utility checks the wrapper utility according to a schedule 710, e.g., every 30 minutes, once daily, at set times, etc., to determine if the wrapper that manages execution of maintenance tools is running. (The cron utility only makes sure the wrapper is running; the wrapper utility then runs continuously and periodically checks whether maintenance should be executed.) If it is not found to be running, the cron scheduling utility starts the wrapper, which then monitors the need for maintenance in a dynamic, continuous fashion. The wrapper then determines whether the maintenance tools are already running 712. If the maintenance tools are actively executing currently, the wrapper will not start parallel maintenance tasks. Next, the need for maintenance is ascertained through checking current conditions such as free disk space 714. A determination is made whether the maintenance utility should be performed now 720 (for example, if maintenance tasks have not yet been performed today, if free disk space is getting too low, etc.). If not 724, the wrapper utility sleeps again for a predetermined amount of time 740, after which the process is repeated. If the maintenance utility should be run now 722, the maintenance utility is performed 730.

The process illustrated with reference to Figs. 1-7 may be tangibly embodied in a computer-readable medium or carrier, e.g. one or more of the fixed and/or removable

data storage devices 388 illustrated in Fig. 3, or other data storage or data communications devices. The computer program 390 may be loaded into memory 312 to configure the system 300 for execution of the computer program 390. The computer program 390 include instructions which, when read and executed by a processor, such as central processing unit 310 of Fig. 1, causes the devices to perform the steps necessary to execute the steps or elements of an embodiment of the present invention.

Accordingly, embodiments of the present invention ensure that the servers are operating according to specifications. Referring again to Fig. 3, the maintenance tool 352 periodically performs maintenance tasks via control of the maintenance scheduling device 350, and if any of the tasks cannot be accomplished, or if an error is detected, the maintenance tool alerts the system administrator. The frequency that maintenance scheduling device 350 causes the maintenance tool 352 to run is determined according to predetermined criteria, e.g., timing, system conditions such as disk space availability, server usage, etc.

The foregoing description of the exemplary embodiment of the invention has been presented for the purposes of illustration and description. It is not intended to be exhaustive or to limit the invention to the precise form disclosed. Many modifications and variations are possible in light of the above teaching. It is intended that the scope of the invention be limited not with this detailed description, but rather by the claims appended hereto.